

# PATENT APPLICATION

## REAL TIME TRANSPORT PROTOCOL CONNECTOR

INVENTORS: Marc Overfeldt  
651 Azara Place #1  
Sunnyvale, CA 94086  
Citizen of Germany

Ivan Wong  
1285 Littleton Drive  
San Jose, CA 95131  
Citizen of United States

Michael Bundschuh  
2199 Cedar Ave.  
Menlo Park, CA 94025  
Citizen of United States

ASSIGNEE: Sun Microsystems, Inc.  
901 San Antonio Road  
Palo Alto, CA 94303

MARTINE & PENILLA, LLP  
710 Lakeway Drive, Suite 170  
Sunnyvale, CA 94085  
Telephone (408) 749-6900

# REAL TIME TRANSPORT PROTOCOL CONNECTOR

*by Inventors*

5

*Marc Owerfeldt, Ivan Wong, and Michael Bundschuh*

## **BACKGROUND OF THE INVENTION**

### **1. Field of the Invention**

The present invention relates generally to real-time multimedia streaming, and  
10 more particularly to a transport-independent real-time protocol stack.

### **2. Description of the Related Art**

With the explosive growth of the Internet, there is a growing interest in using the  
Internet and other Internet protocol-based networks to deliver multimedia selections, such  
as audio and video material. Scalable, open-architecture multimedia systems are being  
15 used to store and retrieve multimedia data over the Internet. Interactive television,  
movies on demand, and other multimedia technologies are among the more promising  
applications for use on these systems.

The Internet is a wide area network offering best effort delivery service. Packets  
of data are routed as datagrams that carry the address of the intended recipient. A specific  
20 connection between the sender and the recipient is not required because all the host nodes  
on the network include the inherent capability to route datagrams from node to node until  
delivery is effected. This datagram packet delivery scheme is constructed as a best effort  
delivery system in which the delivery of datagram packets is not guaranteed.

In many cases, multimedia data requires real-time delivery. In the case of audio or video data, the data stream representing a particular media selection needs to be delivered in the proper sequence and within an abbreviated time period, to allow the user to play back the audio or video selection as it is being sent. The Real-Time Transport Protocol (RTP) is a current de facto standard for delivering real-time content over the Internet or other networks. In the case of an Internet Protocol (IP) based network such as the Internet, RTP utilizes the User Datagram Protocol (UDP) over IP for transport. The term RTP generally refers to two complementary protocols, RTP and Real-Time Control Protocol (RTCP), both defined in RFC 1889: "A Transport Protocol for Real-Time Applications," which is incorporated herein by reference. The RTP specifies how to carry data that has real-time properties, while the RTCP monitors the quality of service (QoS) and conveys information concerning the participants in an on-going session.

While RTP provides a framework for delivering multimedia streaming data over computer networks, conventional RTP implementations are transport specific. Figure 1 is a block diagram showing a conventional RTP based multimedia system 100. The multimedia system 100 includes a multimedia application 102 having a transport specific RTP stack 104, which facilitates streaming via a network 106, such as the Internet. Conventional RTP stacks 104 are designed and implemented for a specific network, for example, an Internet Protocol (IP) based network such as the Internet 106. These implementations intermix transport-dependent tasks, such as IP address management, with transport-independent elements, such as packet decoders and encoders.

Unfortunately, this leads to extremely complex systems that cannot easily be ported or adapted to other transport mechanisms, for example Asynchronous Transfer

Mode (ATM) based networks. In addition, the inherent complexity of these transport specific RTP stacks 104 makes them error prone and difficult to maintain and extend.

In view of the foregoing, there is a need for an RTP stack that is transport-independent. In addition, the RTP stack should be easily adapted to operate with

5 fundamentally different types of transport layers.

11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2

## SUMMARY OF THE INVENTION

Broadly speaking, the present invention fills these needs by providing a transport-independent real-time protocol stack. The present invention provides an RTP connector that separates transport-independent tasks from transport-dependent tasks, thus allowing the real-time protocol stack to be easily adapted to operate with different transport layers. In one embodiment, a transport-independent RTP stack is disclosed. The transport-independent RTP stack includes a transport-independent tasks module, which includes methods that are independent of an underlying transport layer. In communication with the transport-independent module is a connector module, which includes methods that are dependent on the underlying transport layer.

In another embodiment, an RTP connector module is disclosed. The RTP connector module includes an RTP output stream method that returns an RTP output stream to a calling method, and an RTP input stream method that returns an RTP input stream to a calling method. In addition, the RTP connector module includes an RTCP output stream method that returns an RTCP output stream to a calling method, and an RTCP input stream method that returns an RTCP input stream to a calling method.

A further transport-independent RTP stack is disclosed in another embodiment of the present invention. The transport-independent RTP stack includes a transport-independent tasks module having an RTP transmitter module and an RTP receiver module that are each independent of a first underlying transport layer. In addition, the transport-independent RTP stack includes a connector module. The connector module has an RTP output stream method, which is in communication with the RTP transmitter module, and an RTP input stream method, which is in communication with the RTP

receiver module. The RTP output stream method and the RTP input stream provide access to the first underlying transport layer. Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

FIG. 1  
FIG. 2  
FIG. 3  
FIG. 4  
FIG. 5  
FIG. 6  
FIG. 7  
FIG. 8  
FIG. 9  
FIG. 10  
FIG. 11  
FIG. 12  
FIG. 13  
FIG. 14  
FIG. 15  
FIG. 16  
FIG. 17  
FIG. 18  
FIG. 19  
FIG. 20  
FIG. 21  
FIG. 22  
FIG. 23  
FIG. 24  
FIG. 25  
FIG. 26  
FIG. 27  
FIG. 28  
FIG. 29  
FIG. 30  
FIG. 31  
FIG. 32  
FIG. 33  
FIG. 34  
FIG. 35  
FIG. 36  
FIG. 37  
FIG. 38  
FIG. 39  
FIG. 40  
FIG. 41  
FIG. 42  
FIG. 43  
FIG. 44  
FIG. 45  
FIG. 46  
FIG. 47  
FIG. 48  
FIG. 49  
FIG. 50  
FIG. 51  
FIG. 52  
FIG. 53  
FIG. 54  
FIG. 55  
FIG. 56  
FIG. 57  
FIG. 58  
FIG. 59  
FIG. 60  
FIG. 61  
FIG. 62  
FIG. 63  
FIG. 64  
FIG. 65  
FIG. 66  
FIG. 67  
FIG. 68  
FIG. 69  
FIG. 70  
FIG. 71  
FIG. 72  
FIG. 73  
FIG. 74  
FIG. 75  
FIG. 76  
FIG. 77  
FIG. 78  
FIG. 79  
FIG. 80  
FIG. 81  
FIG. 82  
FIG. 83  
FIG. 84  
FIG. 85  
FIG. 86  
FIG. 87  
FIG. 88  
FIG. 89  
FIG. 90  
FIG. 91  
FIG. 92  
FIG. 93  
FIG. 94  
FIG. 95  
FIG. 96  
FIG. 97  
FIG. 98  
FIG. 99  
FIG. 100

## **BRIEF DESCRIPTION OF THE DRAWINGS**

The invention, together with further advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

5        Figure 1 is a block diagram showing a conventional RTP based multimedia system;

Figure 2 is a diagram showing multimedia system, in accordance with an embodiment of the present invention;

Figure 3 is a block diagram showing an RTP packet for use in a UDP/IP stack;

10       Figure 4 is a block diagram showing an RTP based multimedia system, in accordance with an embodiment of the present invention;

Figure 5 is a block diagram showing a transport-independent RTP stack, in accordance with an embodiment of the present invention;

15       Figure 6A is a block diagram showing an RTP connector, in accordance with an embodiment of the present invention;

Figure 6B is a block diagram showing an RTP connector having additional functionality, in accordance with another embodiment of the present invention;

Figure 7 is a block diagram illustrating internal methods of a transport-independent RTP stack, in accordance with an embodiment of the present invention; and

Figure 8 is a block diagram of an exemplary computer system for carrying out the processing according to the invention.

Figure 8 is a block diagram of an exemplary computer system for carrying out the processing according to the invention.



## **DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS**

An invention is disclosed for a transport-independent real-time protocol stack. The present invention provides an RTP connector that separates transport-independent tasks from transport-dependent tasks. In addition, the RTP connector of the embodiments  
5 of the present invention allows the RTP stack to be easily adapted to any type of transport layer. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps have not been  
10 described in detail in order not to unnecessarily obscure the present invention.

Figure 1 was described in terms of the prior art. Figure 2 is a diagram showing multimedia system 200, in accordance with an embodiment of the present invention. The multimedia system 200 includes a multimedia application 202 and illustrates a plurality of sources from which the multimedia application 202 can receive data. These sources  
15 include a local drive 204, a video camera 206, and a network 208, such as the Internet. When the source is a network 208, the transport protocol can vary, for example, the data can be received using HTTP 210 or RTP 212.

When the data source is in direct communication with the multimedia application, such as in the case of the local drive 204 or the video camera 206, it is a relatively simple  
20 matter to receive, decode, and process the data. However, when multimedia data is received via a network 208, the mechanics of receiving and processing the data packets is more complex. Although HTTP is a generally accepted protocol for use in transmitting

non-real-time data such as web pages, HTTP 210 generally does not perform well when used for real-time data transmission. In these situations, RTP 212 generally is used for receiving multimedia data packets.

RTP is a protocol that supports real-time transmission of voice and video. Figure 3 is a block diagram showing an RTP packet 300 for use in a UDP/IP stack. The RTP packet 300 includes an Ethernet header 302, an IP header 304, a User Datagram Protocol (UDP) header 306, an RTP header 308, data 310, and an Ethernet trailer 312. Although Figure 3 illustrates an RTP packet 300 that operates using a UDP/IP stack, it should be noted that the embodiments of the present invention can be used with any type of stack. It should also be noted that the RTP packet 300 can be used with any type of layer 2 protocol. For exemplary purposes the RTP packet 300 is shown for use with the Ethernet protocol. When used in a UDP/IP stack, the RTP packet 308 generally is created first and then the packet is moved down the stack to UDP and IP.

The RTP packet 300 rides on top of UDP and includes timestamping and synchronization information in the RTP header 308 for proper reassembly at the receiving end. RTCP is a companion protocol that is used to maintain Quality of Service (QoS). RTP transmitters, receivers, mixers and translators periodically send each other RTCP packets that include information such as timestamp correlations used to synchronize streams, interarrival jitter, and transmission delays, which allow analysis of the condition of the network and the quality of server, for example, by computing round-trip times.

As mentioned above, conventional RTP stacks are transport-specific, leading to difficulties when the application needs to be ported into another environment. The

embodiments of the present invention address this issue by providing a transport-independent RTP stack.

Figure 4 is a block diagram showing an RTP based multimedia system 400, in accordance with an embodiment of the present invention. The multimedia system 400 includes a multimedia application 402 having a transport-independent RTP stack 404, which facilitates streaming via the network 106, such as the Internet. As mentioned above, conventional RTP stacks are designed and implemented for a specific network, for example, an IP-based network such as the Internet. These implementations intermix transport-dependent tasks, such as IP address management, with transport-independent elements, such as packet decoders and encoders.

However, the embodiments of the present invention provide a transport-independent RTP stack 404, which can be easily adapted to any type of transport layer, such as ATM or IP transport layers. In particular, the transport-independent RTP stack 404 of the embodiments of the present invention separates truly transport-independent tasks from the transport-dependent tasks. As will be discussed in greater detail subsequently, the embodiments of the present invention also provide an RTP connector that can be used to adapt the transport-independent RTP stack 404 to any type of transport layer.

In this manner, the multimedia application 402 can be written in generic terms to source and sink streaming data over the network 106 via the transport-independent RTP stack 404. Thereafter, the multimedia application 402 can be adapted to operate over another network environment by adapting the RTP connector of the transport-independent RTP stack 404 to the new network transport layer. Thus, the embodiments

of the present invention allow the RTP stack 404, and by extension the utilizing application 402, to be easily ported to any type of transport layer without extensive re-writing of the application code.

Figure 5 is a block diagram showing a transport-independent RTP stack 404, in accordance with an embodiment of the present invention. The transport-independent RTP stack 404 includes a transport-independent tasks module 500 and an RTP connector 502, which facilitates communication with the transport layer 504. The transport-independent tasks module 500 includes the truly transport-independent tasks utilized during RTP streaming. For example, these transport-independent tasks can include packetization, depacketization, session management, and other transport-independent tasks as will be apparent to those skilled in the art.

The RTP connector 502 includes the transport-dependent tasks utilized during RTP streaming via the transport layer 504. These transport-dependent tasks can include transmission tasks, data receiving tasks, and other transport-dependent tasks as will be apparent to those skilled in the art. As mentioned above, the RTP connector 502 can be used to adapt the transport-independent RTP stack 404 to any type of transport layer.

In particular, the RTP connector 502 can be implemented to operate with any type of transport 504, then registered using an initialization call. Although the following discussion describes the present invention in terms of the Java programming language, it should be noted that any programming language can be used to implement the embodiments of the present invention. For example, when implementing the embodiments of the present invention using Java, the Java classes are compiled into machine independent bytecode class files which are executed by a machine-dependent

virtual machine. The virtual machine provides a level of abstraction between the machine independence of the bytecode classes and the machine-dependent instruction set of the underlying computer hardware. A class loader is responsible for loading the byte-code class files as needed, and an interpreter or just-in-time compiler provides for the transformation of bytecodes into machine code.

More specifically, Java is a programming language designed to generate applications that can run on all hardware platforms, small, medium and large, without modification. Developed by Sun, Java has been promoted and geared heavily for the Web, both for public Web sites and Intranets. Generally, Java programs can be called from within HTML documents or launched standalone. When a Java program runs from a Web page, it is called a "Java applet," and when run on a Web server, the application is called a "servlet."

Java is an interpreted language. The source code of a Java program is compiled into an intermediate language called "bytecode." The bytecode is then converted (interpreted) into machine code at runtime. Upon finding a Java applet, the Web browser invokes a Java interpreter (Java Virtual Machine), which translates the bytecode into machine code and runs it. Thus, Java programs are not dependent on any specific hardware and will run in any computer with the Java Virtual Machine software. On the server side, Java programs can also be compiled into machine language for faster performance. However a compiled Java program loses hardware independence as a result.

As mentioned above, in addition to Java, other programming languages may be used to implement the embodiments of the present invention, including both procedural

and object oriented programming languages. Object-oriented programming is a method of creating computer programs by combining certain fundamental building blocks, and creating relationships among and between the building blocks. The building blocks in object-oriented programming systems are called "objects." An object is a programming unit that groups together a data structure (instance variables) and the operations (methods) that can use or affect that data. Thus, an object consists of data and one or more operations or procedures that can be performed on that data. The joining of data and operations into a unitary building block is called "encapsulation."

An object can be instructed to perform one of its methods when it receives a "message." A message is a command or instruction to the object to execute a certain method. It consists of a method selection (name) and a plurality of arguments that are sent to an object. A message tells the receiving object what operations to perform.

One advantage of object-oriented programming is the way in which methods are invoked. When a message is sent to an object, it is not necessary for the message to instruct the object how to perform a certain method. It is only necessary to request that the object execute the method. This greatly simplifies program development.

Object-oriented programming languages are predominantly based on a "class" scheme. A class defines a type of object that typically includes both instance variables and methods for the class. An object class is used to create a particular instance of an object. An instance of an object class includes the variables and methods defined for the class. Multiple instances of the same class can be created from an object class. Each instance that is created from the object class is said to be of the same type or class.

A hierarchy of classes can be defined such that an object class definition has one or more subclasses. A subclass inherits its parent's (and grandparent's etc.) definition. Each subclass in the hierarchy may add to or modify the behavior specified by its parent class.

5 Thus, using the Java language, embodiments of the present invention can provide a default RTP connector 502 for use with a particular type of transport, for example, an IP-based network via UDP datagram packets. Then, a new RTP connector 502 can be designed as a class for use over another transport, such as ATM. The new ATM RTP connector class can then be implemented by passing the new ATM RTP connector class  
10 to an initialization method.

The transport-independent tasks module 500 acts as a data source and data sink for the application it is servicing. In this manner, the application program using the transport-independent RTP stack 404 can be designed to communicate with the transport-independent tasks module 500 without regard to the specific type of network transport  
15 protocol that will be used with the system. Similarly, the transport-independent tasks module 500 communicates with the RTP connector 502 without regard to the specific protocol used for the transport 504. In particular, the transport-independent tasks module 500 sends data to, and receives data from the RTP connector 502 in the same manner, regardless of the specific protocol used for the transport 504.

20 Figure 6A is a block diagram showing an RTP connector 502a, in accordance with an embodiment of the present invention. The RTP connector 502a includes four fundamental methods for reading from and writing to the transport layers. Specifically, RTP connector 502a includes an RTP output stream method 600, an RTP input stream

method 602, an RTCP output stream method 604, and an RTCP input stream method 606.

The RTP output stream method 600 returns an output stream to send RTP data out over the network via the transport layers. To write data, a Java program opens a stream to  
5 a data sink and writes to it in a serial fashion. Using the RTP output stream method 600, the transport-independent tasks module 500 can write RTP data to the network regardless of the actual transport being used.

The RTP input stream method 602 returns an input stream to receive RTP data from the network via the transport layers. To bring data into a program, a Java program  
10 opens a stream to a data source and reads the information serially. Similar to the RTP output stream method 600, the transport-independent tasks module 500 can read RTP data from the network regardless of the actual transport being used via the RTP input stream method 602.

The RTCP output stream method 604 returns an output stream to send RTCP data  
15 out over the network via the transport layers, and the RTCP input stream method 606 returns an input stream to receive RTCP data from the network via the transport layers. As with the RTP streams, the RTCP output stream method 604 and the RTCP input stream method 606 can be used to read and write RTCP control data to and from the network regardless of the actual transport protocol being used.

20 In addition to the fundamental methods described above, an RTP connector of the embodiments of the present invention can include additional transport-dependent methods that are useful for managing transport input/output. Figure 6B is a block



diagram showing an RTP connector 502b having additional functionality, in accordance with another embodiment of the present invention. The RTP connector 502b includes a plurality of supplementary methods in addition to the RTP and RTCP stream methods discussed above with respect to Figure 6A.

5           These supplementary methods include a close all RTP/RTCP streams method 608, a get receive buffer size method 608, a get send buffer size method 612, a set receive buffer size method 614, a return RTCP bandwidth fraction method 616, a set send buffer size method 618, and a return RTCP sender bandwidth fraction method 620. The set receive buffer size method 614 is used by the platform's networking code as a hint for the  
10 size to set the underlying network I/O buffers. The get receive buffer size method 608 returns the value that is the buffer size used by the platform for input.

          The set send buffer size method 618 specifies the desired buffer size, which provides a hint to the platform's networking code as to the size to set the underlying network I/O buffers. Increasing the buffer size can enhance the performance of the  
15 network I/O for high-volume connection, while decreasing the buffer size can help reduce the backlog of incoming data. For UDP, the set send buffer size method 618 sets a maximum size of a packet that can be sent on the socket. Conversely, the get send buffer size method 612 returns the buffer size that is used by the platform for output.

          The return RTCP bandwidth fraction method 616 returns the bandwidth portion  
20 utilized for RTCP, which is generally about 5 percent of the total bandwidth available. The RTCP bandwidth fraction can also be used to initialize the RTPManager object. The RTP specification recommends  $\frac{1}{4}$  of the RTCP bandwidth to be dedicated to active data

senders. The return RTCP sender bandwidth fraction method 620 returns the sender bandwidth portion, and can also be used to initialize the RTPManager object.

The RTP connector 502a of the embodiments of the present invention reduces the transport-dependent tasks to a small set of methods. Then, the data that is generated by these methods can be processed in general terms, rather than in complex, network-specific terms, as discussed in greater detail next with reference to Figure 7.

Figure 7 is a block diagram illustrating internal methods of a transport-independent RTP stack 404, in accordance with an embodiment of the present invention. The transport-independent RTP stack 404 includes a transport-independent tasks module 500 and an RTP connector 502. The transport-independent tasks module 500 includes the truly transport-independent tasks utilized during RTP streaming. For example, these transport-independent tasks can include packetization, depacketization, session management. In addition, the transport-independent tasks module 500 includes a plurality of RTP/RTCP modules, namely, an RTP transmitter module 700, an RTP receiver module 702, an RTCP transmitter module 704, and an RTCP receiver module 706.

The RTP connector 502 includes the transport-dependent tasks utilized during RTP streaming via the transport. As mentioned previously, the transport-dependent tasks include four fundamental methods for reading from and writing to the transport layers. Specifically, RTP connector 502 includes an RTP output stream method 600, an RTP input stream method 602, an RTCP output stream method 604, and an RTCP input stream method 606. As mentioned above, RTP connector can be used to adapt the transport-independent RTP stack 404 to any type of transport layer.

In operation, the RTP transmitter module 700 transmits RTP data to the network regardless of the actual transport being used via the RTP output stream method 600. In particular, the RTP output stream method 600 returns an output stream to the RTP transmitter module 700, which can then use the output stream to transmit RTP data to the network in a transport-independent manner, rather than using complex, network-specific operations.

Similarly, the RTP receiver module 702 receives RTP data from the network regardless of the actual transport being used via the RTP input stream method 602. In particular, the RTP input stream method 602 returns an input stream to the RTP receiver module 702, which can then use the input stream to receive RTP data from the network in a transport-independent manner, rather than using complex, network-specific operations.

As with the RTP modules, the RTCP transmitter module 704 and RTCP receiver module 706 transmit and receive RTP data to and from the network regardless of the actual transport being used via the RTCP output stream method 604 and the RTCP input stream method 606. In particular, the RTCP output stream method 604 and the RTCP input stream method 606 return streams to the RTCP transmitter module 704 and the RTCP receiver module 706. The RTCP transmitter module 704 and the RTCP receiver module 706 can then use the streams to transmit and receive RTCP data to and from the network in a transport-independent manner.

Embodiments of the present invention may employ various computer-implemented operations involving data stored in computer systems to drive computer software, including application programs, operating system programs, peripheral device drivers, etc. These operations are those requiring physical manipulation of physical

quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. Further, the manipulations performed are often referred to in terms, such as producing, identifying, determining, or comparing.

5 Any of the operations described herein that form part of the invention are useful machine operations. The invention also relates to a device or an apparatus for performing these operations. The apparatus may be specially constructed for the required purposes, or it may be a general purpose computer selectively activated or configured by a computer program stored in the computer. In particular, various general purpose machines may be  
10 used with computer programs written in accordance with the teachings herein, or it may be more convenient to construct a more specialized apparatus to perform the required operations. An exemplary structure for the invention is described below.

Figure 8 is a block diagram of an exemplary computer system 800 for carrying out the processing according to the invention. The computer system 800 includes a digital  
15 computer 802, a display screen (or monitor) 804, a printer 806, a floppy disk drive 808, a hard disk drive 810, a network interface 812, and a keyboard 814. The digital computer 802 includes a microprocessor 816, a memory bus 818, random access memory (RAM) 820, read only memory (ROM) 822, a peripheral bus 824, and a keyboard controller (KBC) 826. The digital computer 802 can be a personal computer (such as an IBM  
20 compatible personal computer, a Macintosh computer or Macintosh compatible computer), a workstation computer (such as a Sun Microsystems or Hewlett-Packard workstation), or some other type of computer.

The microprocessor 816 is a general purpose digital processor, which controls the operation of the computer system 800. The microprocessor 816 can be a single-chip processor or can be implemented with multiple components. Using instructions retrieved from memory, the microprocessor 816 controls the reception and manipulation of input data and the output and display of data on output devices.

The memory bus 818 is used by the microprocessor 816 to access the RAM 820 and the ROM 822. The RAM 820 is used by the microprocessor 816 as a general storage area and as scratch-pad memory, and can also be used to store input data and processed data. The ROM 822 can be used to store instructions or program code followed by the microprocessor 816 as well as other data.

The peripheral bus 824 is used to access the input, output, and storage devices used by the digital computer 802. In the described embodiment, these devices include the display screen 804, the printer device 806, the floppy disk drive 808, the hard disk drive 810, and the network interface 812. The keyboard controller 826 is used to receive input from keyboard 814 and send decoded symbols for each pressed key to microprocessor 816 over bus 828.

The display screen 804 is an output device that displays images of data provided by the microprocessor 816 via the peripheral bus 824 or provided by other components in the computer system 800. The printer device 806, when operating as a printer, provides an image on a sheet of paper or a similar surface. Other output devices such as a plotter, typesetter, etc. can be used in place of, or in addition to, the printer device 806.

The floppy disk drive 808 and the hard disk drive 810 can be used to store various types of data. The floppy disk drive 808 facilitates transporting such data to other computer systems, and hard disk drive 810 permits fast access to large amounts of stored data.

5 The microprocessor 816, together with an operating system, operates to execute computer code and produce and use data. The computer code and data may reside on the RAM 820, the ROM 822, or the hard disk drive 810. The computer code and data could also reside on a removable program medium and loaded or installed onto the computer system 800 when needed. Removable program media include, for example, CD-ROM,  
10 PC-CARD, floppy disk and magnetic tape.

The network interface 812 is used to send and receive data over a network connected to other computer systems. An interface card or similar device and appropriate software implemented by the microprocessor 816 can be used to connect the computer system 800 to an existing network and transfer data according to standard protocols.

15 The keyboard 814 is used by a user to input commands and other instructions to the computer system 800. Other types of user input devices can also be used in conjunction with the present invention. For example, pointing devices such as a computer mouse, a track ball, a stylus, or a tablet can be used to manipulate a pointer on a screen of a general-purpose computer.

20 The invention can also be embodied as computer readable code on a computer readable medium. The computer readable medium is any data storage device that can store data, which can be thereafter, be read by a computer system. Examples of the

computer readable medium include read-only memory (ROM), random-access memory (RAM), CD-ROMs, magnetic tape, and optical data storage devices. The computer readable medium can also be distributed over a network that couples computer systems so that the computer readable code is stored and executed in a distributed fashion.

5 Furthermore, the same or similar methods and apparatuses described above for programming a hardware device can also be used for performing other particular maintenance operations on the hardware device. For example, operations such as erasing a ROM, reading a ROM, or performing a checksum on a ROM can be performed.

10 Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.

15 ***What is claimed is:***